



**UNIVERSIDADE
KIMPA VITA**

15 - 10 | 2026

Inteligência artificial aplicada à engenharia de software

Artificial intelligence applied to software engineering

Mandongá Fernando Manuel Vuia

Versão electrónica

URL: <https://ciencia.unikivi.ao/>

Data de publicação: 15-10-2026. Páginas: 10

Editor

Revista Científica Interdisciplinar da UNIKIVI

Referência electrónica

Vuia, M. F. M. (2026). Inteligência artificial aplicada à engenharia de software. Revista da UNIKIVI. 02(01). 01-10.



INTELIGÊNCIA ARTIFICIAL APLICADA À ENGENHARIA DE SOFTWARE

Artificial intelligence applied to software engineering

Mandongia Fernando Manuel Vuia

Universidade Kimpa Vita, Angola

mandongafernando@gmail.com | <https://orcid.org/0009-0006-9532-3380>

RESUMO

A Inteligência Artificial (IA) tem assumido importância crescente na Engenharia de Software, com aplicações na Engenharia de Requisitos, desenvolvimento, análise de código, testes, detecção e previsão de defeitos, manutenção, reparação de programas e gestão de projectos. A expansão da IA Generativa e dos Grandes Modelos de Linguagem (*Large Language Models* — LLMs) ampliou as possibilidades de automação e apoio à tomada de decisão ao longo do ciclo de vida do software. O presente artigo analisa as principais aplicações da IA na Engenharia de *Software*, considerando os seus contributos, limitações, desafios e tendências. Adoptou-se uma abordagem qualitativa, exploratória e descritiva, baseada numa revisão narrativa e crítica da literatura científica. A pesquisa incidiu sobre publicações disponíveis na *IEEE Xplore*, *Science Direct*, *Springer Link* e *Google Scholar*, privilegiando estudos publicados entre 2000 e 2025, com atenção à literatura recente sobre IA Generativa e LLMs. Os resultados evidenciam aplicações relevantes na geração e transformação de código, automatização de testes, análise de requisitos, detecção de defeitos, manutenção e apoio à gestão de projectos. Contudo, a literatura identifica limitações relacionadas com precisão, segurança, explicabilidade, qualidade dos dados, dependência do contexto, privacidade e necessidade de validação humana. Conclui-se que a principal contribuição da IA para a Engenharia de *Software* reside na automação assistida e na ampliação das capacidades dos profissionais, e não na sua substituição integral. A adopção responsável destas tecnologias requer supervisão humana, mecanismos de validação, avaliação de riscos e integração adequada nos processos de desenvolvimento de software.

Palavras-chave: Inteligência Artificial; Engenharia de Software; *Machine Learning*; Inteligência Artificial Generativa; Grandes Modelos de Linguagem; Automação de Software.

ABSTRACT

Artificial Intelligence (AI) has been playing an increasingly important role in Software Engineering, with applications in Requirements Engineering, development, code analysis, testing, defect detection and prediction, maintenance, program repair, and project management. The rise of Generative AI and Large Language Models (LLMs) has expanded the possibilities for automation and decision support throughout the software lifecycle. This article looks at the main uses of AI in Software Engineering, considering its contributions, limitations, challenges, and trends. A qualitative, exploratory, and descriptive approach was adopted, based on a narrative and critical review of the scientific literature. The research focused on publications available on IEEE Xplore, ScienceDirect, SpringerLink, and Google Scholar, prioritizing studies published between 2000 and 2025, with attention to recent literature on Generative AI and LLMs. The results show relevant applications in code generation and transformation, test automation, requirements analysis, defect detection, maintenance, and project management support. However, the literature points out limitations related to accuracy, security, explainability, data quality, context dependence, privacy, and the need for human validation. It is concluded that the main contribution of AI to Software

Engineering lies in assisted automation and enhancing professionals' capabilities, rather than fully replacing them. Responsible adoption of these technologies requires human oversight, validation mechanisms, risk assessment, and proper integration into software development processes.

Keywords: Artificial Intelligence; Software Engineering; Machine Learning; Generative Artificial Intelligence; Large Language Models; Software Automation.

1 INTRODUÇÃO

A Inteligência Artificial (IA) constitui uma área da computação dedicada ao desenvolvimento de métodos e sistemas capazes de realizar tarefas associadas a capacidades como aprendizagem, reconhecimento de padrões, inferência, previsão, classificação e geração de conteúdos. As bases conceptuais da IA foram desenvolvidas a partir de diferentes abordagens, incluindo perspectivas simbólicas, conexionistas e evolucionárias, que contribuíram para a evolução dos sistemas inteligentes (Barone, 2003; Barreto, 2007; Costa, 1986). Embora a utilização da IA na Engenharia de Software não seja recente, a evolução das técnicas de *Machine Learning* (ML), *Deep Learning* (DL), Processamento de Linguagem Natural (PLN) e, mais recentemente, da IA Generativa e dos Grandes Modelos de Linguagem (*Large Language Models* — LLMs), tem ampliado significativamente o conjunto de actividades susceptíveis de apoio ou automatização (Pacola, 2021).

Na Engenharia de Software, a IA tem sido aplicada em diferentes fases do ciclo de vida, incluindo Engenharia de Requisitos, projecto, implementação, testes, manutenção e gestão. A Engenharia de Software estrutura estas actividades através de processos, métodos e técnicas destinados a assegurar o desenvolvimento e a evolução sistemática de sistemas de software (Carvalho, 2001; Falbo, 2005; Sommerville, 2016). Neste contexto, a utilização de técnicas computacionais inteligentes pode apoiar actividades relacionadas com análise, qualidade, avaliação de riscos e tomada de decisão em projectos de software (Boente, 2009; Castro, 2009). A literatura recente mostra, contudo, que a aplicação destas tecnologias não deve ser compreendida apenas como automatização de tarefas de programação. A investigação tem abrangido igualmente actividades de garantia da qualidade, manutenção, análise de artefactos de software, apoio à decisão e outras tarefas de engenharia (Kokol, 2024).

A expansão da IA Generativa introduziu uma nova dimensão neste domínio. Ferramentas baseadas em LLMs podem apoiar a geração, explicação, transformação e documentação de código, bem como determinadas actividades de testes, requisitos e manutenção. A utilização da IA no desenvolvimento de software enquadra-se, assim, numa evolução mais ampla das ferramentas computacionais de apoio às actividades de Engenharia de Software (Pacola, 2021). Uma revisão sistemática dedicada aos LLMs em Engenharia de Software analisou centenas de estudos e evidenciou a amplitude crescente desta área de investigação (Hou et al., 2024).

A relação entre conhecimento especializado e desenvolvimento de software constitui igualmente um aspecto relevante. A experiência dos profissionais e o conhecimento do domínio influenciam a forma como os problemas de software são compreendidos e como as soluções são concebidas, aspecto particularmente importante quando ferramentas automatizadas passam a participar em actividades anteriormente realizadas predominantemente por especialistas (Adelson & Soloway, 1985). Neste sentido, a utilização da IA deve ser compreendida como uma forma de apoio às actividades de Engenharia de Software, sem eliminar a necessidade de conhecimento técnico e de avaliação humana.

Apesar destas possibilidades, a adopção da IA não significa que as actividades tradicionais da Engenharia de Software deixem de ser necessárias. Pelo contrário, questões relacionadas com validação, segurança, qualidade, rastreabilidade, dependabilidade e responsabilidade tornam necessária a intervenção de profissionais capazes de avaliar criticamente as saídas produzidas pelos sistemas de IA. A qualidade do software e a gestão dos riscos constituem aspectos fundamentais dos processos de desenvolvimento, particularmente em contextos nos quais decisões e resultados produzidos por sistemas computacionais podem afectar o desempenho dos projectos (Boente, 2009; Castro, 2009). Estudos recentes sobre IA generativa em Engenharia de Software salientam precisamente a necessidade de investigação adicional sobre precisão,

transparência, acessibilidade dos dados, sustentabilidade e dependabilidade (Nguyen-Duc et al., 2025).

A adopção de tecnologias de IA também se relaciona com transformações mais amplas nas actividades profissionais e na organização do trabalho. A evolução da IA tem suscitado discussões sobre a possibilidade de substituição, transformação ou complementaridade de determinadas actividades humanas, particularmente quando sistemas inteligentes passam a executar tarefas anteriormente realizadas por profissionais (Angeli, 2019). Na Engenharia de Software, esta questão deve ser analisada considerando não apenas a capacidade de automatização, mas também a necessidade de supervisão, conhecimento especializado, validação e responsabilidade profissional.

Neste contexto, a utilização da IA apresenta simultaneamente oportunidades e desafios. Por um lado, pode reduzir o esforço associado a determinadas tarefas repetitivas, apoiar a análise de grandes volumes de informação e contribuir para a automatização assistida. Por outro, pode introduzir resultados incorrectos, vulnerabilidades, problemas de privacidade e dependência excessiva das ferramentas. A avaliação dos riscos associados aos projectos de software e a procura de mecanismos que contribuam para a qualidade dos produtos constituem, por isso, questões relevantes para a adopção responsável de tecnologias computacionais inteligentes (Boente, 2009; Castro, 2009).

Deste problema decorre a seguinte questão de investigação:

Como têm sido aplicadas as técnicas de Inteligência Artificial nas diferentes fases da Engenharia de Software e quais os principais benefícios, limitações e tendências identificados na literatura científica?

O objectivo geral do estudo consiste em analisar as principais aplicações da Inteligência Artificial na Engenharia de Software, considerando os seus contributos, limitações e tendências de desenvolvimento.

Especificamente, pretende-se: i) Identificar as principais técnicas de IA aplicadas à Engenharia de Software; ii) Caracterizar as aplicações da IA nas diferentes fases do ciclo de vida do software; iii) Analisar os principais benefícios associados à utilização da IA; iv) Identificar limitações, riscos e desafios da sua adopção; v) Analisar as principais tendências relacionadas com IA Generativa e LLMs na Engenharia de Software.

O artigo encontra-se organizado em seis secções principais: introdução, enquadramento teórico, metodologia, resultados, discussão e conclusão.

2 ENQUADRAMENTO TEÓRICO

2.1 Inteligência Artificial

A Inteligência Artificial pode ser entendida como o domínio da computação dedicado ao desenvolvimento de sistemas capazes de executar tarefas que envolvem diferentes formas de aprendizagem, raciocínio, reconhecimento, previsão ou tomada de decisão (Russell & Norvig, 2021).

A evolução da IA deu origem a diferentes abordagens, incluindo sistemas baseados em regras, aprendizagem automática, redes neuronais, aprendizagem profunda, processamento de linguagem natural e, mais recentemente, modelos generativos e LLMs.

Na Engenharia de Software, estas técnicas podem ser utilizadas para analisar artefactos, identificar padrões, produzir previsões e apoiar determinadas decisões. A sua utilização, contudo, deve ser relacionada com o problema concreto, a qualidade dos dados disponíveis e os requisitos do sistema.

2.2 Machine Learning

Machine Learning corresponde ao conjunto de técnicas que permitem aos sistemas aprender padrões a partir de dados e utilizá-los para realizar tarefas como classificação, previsão ou detecção de anomalias.

Na Engenharia de Software, os modelos de ML podem ser utilizados, entre outras aplicações, para previsão de defeitos, estimativa de esforço, classificação de requisitos, análise de código, previsão de riscos e apoio aos testes.

A utilização de ML depende da disponibilidade de dados representativos e de qualidade. Consequentemente, o desempenho do modelo não deve ser interpretado isoladamente das características dos dados utilizados no seu desenvolvimento e avaliação.

2.3 Deep Learning

O *Deep Learning* baseia-se em redes neuronais com múltiplas camadas capazes de aprender representações a partir dos dados. A técnica tem sido aplicada em problemas relacionados com reconhecimento de padrões, processamento de linguagem, análise de código e outros artefactos de software. Na Engenharia de Software, as aplicações incluem previsão de defeitos, localização de falhas, análise de código, geração de conteúdos e apoio a determinadas tarefas de manutenção.

2.4 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) permite o processamento computacional da linguagem humana. A sua importância na Engenharia de Software resulta do facto de muitos artefactos, particularmente os requisitos, a documentação e os comentários do código, serem expressos através de linguagem natural.

As técnicas de PLN podem apoiar a classificação e extracção de requisitos, identificação de ambiguidades, sumarização de documentos, geração de documentação e pesquisa de informação relacionada com projectos de software.

2.5 IA Generativa e Grandes Modelos de Linguagem

A IA Generativa refere-se a sistemas capazes de produzir novos conteúdos a partir de instruções ou dados de entrada. No domínio da Engenharia de Software, os LLMs passaram a ser utilizados para geração, explicação, transformação e documentação de código, bem como para apoiar actividades de requisitos, testes e manutenção.

A investigação recente demonstra que esta área se encontra em rápida expansão. Hou et al. (2024) apresentam uma revisão sistemática dedicada à aplicação de LLMs na Engenharia de Software, enquanto Nguyen-Duc et al. (2025) identificam 78 questões de investigação distribuídas por 11 áreas de Engenharia de Software. Estes autores salientam que a literatura ainda se encontra em desenvolvimento e que a investigação apresenta maior concentração em implementação, garantia da qualidade e manutenção, enquanto áreas como requisitos, projecto e educação em Engenharia de Software necessitam de investigação adicional.

Assim, os LLMs não devem ser considerados apenas como geradores automáticos de código. A sua aplicação deve ser compreendida como uma forma de automação assistida, em que os profissionais utilizam os sistemas para apoiar determinadas tarefas, mantendo a responsabilidade pela avaliação e validação das respectivas saídas.

2.6 Aplicações da inteligência artificial na engenharia de software

2.6.1 Engenharia de Requisitos

A Engenharia de Requisitos constitui uma das áreas em que a IA pode apoiar a análise de grandes volumes de informação textual. Técnicas de PLN, ML e LLMs podem ser utilizadas para classificar requisitos, extrair informação, identificar ambiguidades, organizar documentos e apoiar a transformação de requisitos expressos em linguagem natural.

A utilização de IA nesta área pode reduzir determinadas tarefas manuais, mas não elimina a necessidade de validação por especialistas. Os requisitos envolvem frequentemente contexto organizacional, restrições técnicas e necessidades dos utilizadores que podem não ser totalmente identificáveis por um modelo automático.

Uma revisão sistemática recente sobre IA generativa na Engenharia de Requisitos analisou 238 artigos publicados entre 2019 e 2025, demonstrando o crescimento significativo desta linha de investigação (Cheng et al., 2025).

2.6.2 *Desenvolvimento e geração de código*

A geração e transformação de código constituem uma das aplicações mais visíveis dos LLMs na Engenharia de Software. Ferramentas baseadas em IA podem produzir fragmentos de código, sugerir implementações, explicar programas existentes, converter código entre linguagens e apoiar a documentação.

Contudo, a produção automática de código não garante, por si só, correcção funcional, segurança ou qualidade. O código produzido deve ser submetido a revisão, testes e validação de acordo com os requisitos do sistema. Deste modo, a principal contribuição identificada consiste na redução do esforço associado a determinadas tarefas de programação, e não na eliminação da responsabilidade do programador.

2.6.3 *Testes de software*

Os testes constituem outra área com aplicações relevantes de IA.

As técnicas de ML, DL e LLMs podem apoiar a geração de casos de teste, selecção e priorização de testes, análise dos resultados, identificação de anomalias e manutenção de testes automatizados. Estas aplicações podem contribuir para a eficiência do processo de qualidade, sobretudo quando existe um grande volume de casos de teste ou de dados históricos.

Contudo, a eficácia depende da qualidade dos dados, dos modelos utilizados e das características do sistema testado. A utilização de IA nos testes deve, por isso, ser acompanhada por critérios de cobertura, validade e adequação ao contexto do software.

2.6.4 *Detecção e previsão de defeitos*

Os modelos de ML podem utilizar dados históricos relacionados com código-fonte, alterações, resultados de testes e registos de defeitos para identificar padrões associados à ocorrência de problemas. A previsão de defeitos pode apoiar a identificação de componentes que merecem maior atenção durante as actividades de desenvolvimento e manutenção.

Todavia, a previsão não deve ser interpretada como uma garantia de que um determinado componente contém ou não contém defeitos. Trata-se de uma forma de apoio à decisão cujo desempenho depende dos dados e do modelo utilizado.

2.6.5 *Manutenção e reparação de software*

A manutenção de software envolve actividades relacionadas com correcção de defeitos, adaptação, melhoria e compreensão de sistemas existentes. A IA pode apoiar a análise de código, geração de documentação, localização de possíveis falhas e sugestão de alterações. Os LLMs podem igualmente apoiar a compreensão de código legado e produzir propostas de correcção.

Apesar destas possibilidades, a reparação automática não deve ser considerada universalmente fiável. Uma alteração aparentemente correcta pode introduzir efeitos secundários ou não satisfazer completamente os requisitos do sistema. Assim, a utilização da IA deve ser acompanhada de testes e revisão humana.

2.6.6 *Gestão de projectos de software*

A IA pode igualmente apoiar actividades de gestão de projectos através da análise de dados históricos, estimativa de esforço, identificação de riscos, acompanhamento de tarefas e apoio à tomada de decisão. Neste domínio, os modelos podem auxiliar a identificação de padrões e fornecer estimativas com base em dados anteriores. Contudo, as decisões de gestão envolvem factores humanos, organizacionais e contextuais que não podem ser reduzidos exclusivamente a previsões algorítmicas.

3 MÉTODOS

3.1 Abordagem e desenho da investigação

O presente estudo adopta uma abordagem qualitativa, de natureza exploratória e descritiva, baseada numa revisão narrativa e crítica da literatura científica. A opção por uma revisão narrativa e crítica justifica-se pelo objectivo de integrar contributos provenientes de diferentes linhas de investigação sobre IA e Engenharia de Software, permitindo identificar aplicações, benefícios, limitações e tendências sem apresentar o estudo como uma revisão sistemática formal.

3.2 Fontes de informação

A pesquisa incidiu sobre publicações disponíveis em bases de dados e repositórios científicos, com particular atenção às seguintes fontes: i) *IEEE Xplore*; ii) *Science Direct*; iii) *Springer Link*; iv) *Google Scholar*. Foram privilegiadas publicações entre 2000 e 2025, sem excluir trabalhos anteriores considerados fundamentais para o enquadramento conceptual.

Considerando a rápida evolução da IA Generativa, foi atribuída particular atenção às publicações de 2023 a 2025, período em que a investigação sobre LLMs e ferramentas generativas passou a ocupar um espaço significativo na Engenharia de Software.

3.3 Estratégia de pesquisa

Foram utilizados descritores em Língua Inglesa, combinados através dos operadores booleanos AND e OR. Entre os principais termos utilizados encontram-se: i) *Artificial Intelligence*; ii) *Machine Learning*; iii) *Deep Learning*; iv) *Generative AI*; v) *Large Language Models*; vi) *Software Engineering*; vii) *Software Testing*; viii) *Automated Program Repair*; ix) *Requirements Engineering*; x) *Software Development Automation*.

A pesquisa procurou identificar estudos relacionados com aplicações, benefícios, limitações, desafios e tendências da IA na Engenharia de Software.

3.4 Critérios de selecção

Foram considerados pertinentes os estudos que: i) Analisassem aplicações de IA na Engenharia de Software; ii) Abordassem ML, DL, PLN, IA Generativa ou LLMs; iii) Apresentassem resultados, análise conceptual, revisão ou evidência empírica relevante; iv) Discutissem aplicações, benefícios, limitações ou desafios relacionados com a utilização da IA.

Foram excluídos documentos comerciais ou promocionais, trabalhos duplicados, estudos sem relação directa com Engenharia de Software e documentos sem informação científica suficiente para responder aos objectivos do estudo.

3.5 Procedimento de análise

A literatura seleccionada foi organizada segundo as principais fases e actividades da Engenharia de Software: i) Engenharia de Requisitos; ii) Desenvolvimento; iii) Análise de Código; iv) Testes; v) Detecção de defeitos; vi) Manutenção e reparação; vii) Gestão de projectos.

Para cada área foram analisados os principais usos da IA, benefícios potenciais, limitações e necessidade de intervenção humana. A análise assumiu natureza temática e interpretativa, procurando identificar padrões e convergências entre os estudos.

4 RESULTADOS

A análise da literatura revelou que a Inteligência Artificial (IA) apresenta aplicações em praticamente todas as principais fases do ciclo de vida do software, abrangendo actividades relacionadas com desenvolvimento, testes, manutenção e gestão. Contudo, verificou-se que o nível de maturidade dessas aplicações é desigual, variando conforme a fase, tecnologia utilizada e finalidade.

4.1 Síntese das aplicações identificadas

Tabela 1

Fase/área	Técnicas predominantes	Principais aplicações	Principais limitações
Engenharia de Requisitos	PLN, ML, LLMs	Classificação, extracção, análise e transformação de requisitos.	Ambiguidade, contexto incompleto e necessidade de validação.
Desenvolvimento	LLMs, ML	Geração, completamento, explicação e transformação de código.	Erros, código inseguro e dependência do contexto.
Análise de código	ML, DL, LLMs	Identificação de padrões, problemas e vulnerabilidades.	Falsos positivos e necessidade de revisão.
Testes	ML, DL, LLMs	Geração, selecção e priorização de testes.	Qualidade variável e necessidade de validação.
Defeitos	ML, DL	Previsão e classificação de defeitos.	Dependência dos dados históricos.
Manutenção	ML, LLMs	Compreensão, documentação e alteração de código.	Contexto limitado e efeitos secundários.
Reparação	LLMs, técnicas de reparação automática	Sugestão ou geração de correcções.	Nem todas as correcções são funcionalmente válidas.
Gestão de projectos	ML, análise preditiva	Estimativas, previsão de riscos e apoio à decisão.	Qualidade dos dados e factores contextuais.

Nota. Síntese conceptual elaborada com base na literatura analisada pelo autor.

Os resultados indicam que a utilização da IA é transversal ao ciclo de vida do software. No entanto, a literatura recente mostra maior concentração em desenvolvimento, implementação, garantia da qualidade e manutenção, enquanto áreas como Engenharia de Requisitos e projecto de software ainda apresentam oportunidades significativas de investigação (Nguyen-Duc et al., 2025).

4.2 Automação assistida

Um dos principais padrões identificados consiste na utilização da IA como mecanismo de automação assistida. Em vez de substituir integralmente o profissional, os sistemas fornecem sugestões, classificações, previsões ou conteúdos que posteriormente são analisados pelo utilizador. Este padrão é particularmente evidente na geração de código, documentação, testes e manutenção.

4.3 Benefícios identificados

Entre os principais benefícios identificados encontram-se: i) Redução do esforço associado a determinadas tarefas repetitivas; ii) Apoio à análise de grandes volumes de dados; iii) Automatização parcial de actividades; iv) Apoio à geração e transformação de código; v) Apoio à identificação de possíveis defeitos; vi) Apoio à documentação; vii) Suporte à tomada de decisão; vii) possibilidade de acelerar determinadas actividades do ciclo de desenvolvimento. Estes benefícios devem, contudo, ser interpretados de acordo com o contexto e não como efeitos universais da adopção da IA.

4.4 Limitações identificadas

A literatura evidencia limitações da IA, incluindo resultados incorrectos, alucinações, riscos de segurança e privacidade, falta de explicabilidade, dependência dos dados, dificuldades contextuais e de avaliação das saídas. Por isso, a sua utilização na Engenharia de Software exige revisão, validação humana e não permite substituir integralmente as actividades profissionais.

5 DISCUSSÃO DE RESULTADOS

Os resultados permitem compreender a IA como uma tecnologia transversal ao ciclo de vida do software, mas com níveis de aplicação e maturidade diferentes entre as várias áreas.

A utilização mais evidente ocorre actualmente no desenvolvimento, na geração de código, nos testes e na manutenção. Esta tendência é coerente com a literatura recente, que identifica maior concentração da investigação nestas áreas. Nguyen-Duc et al. (2025), por exemplo, identificaram uma forte concentração da investigação em implementação, garantia da qualidade e

manutenção, embora também tenham identificado oportunidades de investigação em requisitos, projecto e educação em Engenharia de Software.

A aplicação da IA à Engenharia de Requisitos apresenta igualmente interesse crescente. A utilização de PLN e LLMs pode facilitar a análise de documentos e a identificação de padrões linguísticos. Contudo, os requisitos estão frequentemente associados a conhecimento organizacional e a necessidades específicas dos utilizadores. Consequentemente, a interpretação automática não deve substituir a validação por especialistas.

Na geração de código, a IA pode contribuir para reduzir o esforço associado a determinadas tarefas. Contudo, a existência de uma resposta gerada automaticamente não significa que o código seja correcto, seguro ou adequado ao contexto. A revisão humana, os testes e a análise das dependências continuam a ser necessários.

Os mesmos princípios aplicam-se à reparação automática. A capacidade de gerar uma possível correcção não equivale à garantia de que o defeito foi efectivamente eliminado. Uma alteração pode resolver um caso específico e, simultaneamente, introduzir novos problemas.

No domínio da qualidade, a IA apresenta potencial para apoiar a geração e priorização de testes e a previsão de defeitos. Todavia, os resultados dependem fortemente dos dados utilizados para treinar e avaliar os modelos. Esta dependência constitui uma limitação importante, particularmente quando os dados históricos não representam adequadamente o sistema ou o ambiente actual.

A literatura recente sobre IA Generativa reforça esta necessidade de prudência. Nguyen-Duc et al. (2025) salientam questões relacionadas com dependabilidade, precisão, transparência, acessibilidade dos dados e sustentabilidade. Por sua vez, estudos recentes sobre IA aumentada na qualidade e nos testes destacam problemas como alucinações, limitações na qualidade das respostas, falta de rastreabilidade e possíveis enviesamentos.

Deste modo, os resultados não sustentam uma interpretação segundo a qual a IA substitui o Engenheiro de Software. A evidência disponível aponta antes para uma relação de complementaridade, na qual os sistemas inteligentes podem executar ou apoiar determinadas tarefas, enquanto os profissionais permanecem responsáveis pela definição dos requisitos, avaliação dos resultados, tomada de decisões e garantia da qualidade.

Esta perspectiva é particularmente importante em sistemas críticos, nos quais erros de código, falhas de segurança ou decisões incorrectas podem produzir consequências relevantes. Nesses contextos, a utilização da IA deve ser acompanhada por mecanismos de revisão, testes, rastreabilidade e controlo.

A investigação futura deverá, portanto, ultrapassar a avaliação da capacidade dos modelos para produzir código ou conteúdos e analisar igualmente o impacto da IA na qualidade, segurança, produtividade, organização do trabalho e responsabilidade profissional.

6 LIMITAÇÕES DO ESTUDO

O estudo apresenta quatro limitações principais. Primeiro, constitui uma revisão narrativa e crítica, sem protocolo sistemático de selecção e avaliação quantitativa, resultando numa síntese qualitativa. Segundo, a rápida evolução da IA, sobretudo dos LLMs, pode tornar parte da literatura desactualizada. Terceiro, diferenças metodológicas, bases de dados, ferramentas e critérios de avaliação dificultam comparações directas. Por fim, não foi realizada avaliação experimental própria das ferramentas de IA. Assim, as conclusões decorrem da interpretação da literatura seleccionada, não de experiências conduzidas pelos autores, devendo ser compreendidas no âmbito das limitações metodológicas identificadas.

7 CONCLUSÃO

A análise realizada demonstra que a Inteligência Artificial apresenta aplicações relevantes em diferentes fases da Engenharia de Software, abrangendo Engenharia de Requisitos, desenvolvimento, análise de código, testes, previsão e detecção de defeitos, manutenção, reparação e gestão de projectos.

A evolução recente da IA Generativa e dos Grandes Modelos de Linguagem ampliou estas possibilidades, particularmente na geração e transformação de código, documentação, testes e apoio à manutenção. Contudo, a literatura também evidencia limitações relacionadas com precisão, segurança, qualidade das respostas, dependabilidade, privacidade, transparência e necessidade de validação humana.

Neste sentido, a principal contribuição da IA para a Engenharia de Software não deve ser entendida como substituição integral dos profissionais, mas como ampliação das suas capacidades através de automação assistida, análise de dados e apoio à decisão.

A adoção responsável da IA exige, por conseguinte, integração com práticas tradicionais de Engenharia de Software, incluindo definição rigorosa de requisitos, revisão de código, testes, controlo de qualidade, avaliação de segurança e supervisão humana.

Conclui-se que o desenvolvimento futuro da Engenharia de Software deverá combinar as capacidades dos sistemas de IA com o conhecimento técnico, a experiência, o julgamento profissional e a responsabilidade dos engenheiros. A investigação deverá aprofundar, sobretudo, a avaliação da qualidade, segurança, dependabilidade, transparência e impacto organizacional das ferramentas de IA ao longo do ciclo de vida do software.

8 REFERÊNCIAS

- Adelson, B., & Soloway, E. (1985). The role of domain experience in software design. *IEEE Transactions on Software Engineering*, 11(11), 1351–1360. <https://doi.org/10.1109/TSE.1985.231883>
- Barone, D. (2003). *Sociedades artificiais: A nova fronteira da inteligência nas máquinas*. Bookman.
- Barreto, J. M. (2007). *Inteligência artificial no limiar do século XXI: Abordagem híbrida, simbólica, conexionista e evolucionária*. Unicamp.
- Boente, A. N. (2009). *Um modelo fuzzy para avaliação da qualidade de produtos de software e da satisfação dos gerentes de projectos numa fundação pública estadual* [Dissertação de mestrado, Universidade Estácio de Sá].
- Carvalho, A. M. B. R. (2001). *Introdução à engenharia de software*. Editora da Unicamp.
- Castro, R. D. (2009). *Avaliação de riscos em projectos de software a partir do uso de técnicas de inteligência computacional* [Monografia, Escola Politécnica de Pernambuco, Universidade de Pernambuco].
- Costa, A. C. (1986). *Sobre os fundamentos da inteligência artificial*. CPGCC-UFRGS.
- Falbo, R. A. (2005). *Engenharia de software: Notas de aula*. Universidade Federal do Espírito Santo.
- Cheng, H., Husen, J. H., Lu, Y., Racharak, T., Yoshioka, N., Ubayashi, N., & Washizaki, H. (2026). Generative AI for requirements engineering: A systematic literature review. *Software: Practice and Experience*, 56(2), 141–170. <https://doi.org/10.1002/spe.70029>
- Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., & Wang, H. (2024). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 33(8), Article 220, 1–79. <https://doi.org/10.1145/3695988>
- Kokol, P. (2024). The use of AI in software engineering: A synthetic knowledge synthesis of the recent research literature. *Information*, 15(6), Article 354. <https://doi.org/10.3390/info15060354>
- Nguyen-Duc, A., Cabrero-Daniel, B., Przybylek, A., Arora, C., Khanna, D., Herda, T., Rafiq, U., Melegati, J., Guerra, E., Kemell, K.-K., Saari, M., Zhang, Z., Le, H., Quan, T., & Abrahamsson, P. (2025). Generative artificial intelligence for software engineering—A research agenda. *Software: Practice and Experience*, 55(11), 1806–1843. <https://doi.org/10.1002/spe.70005>
- Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.
- Pacola, V. (2021). *Inteligência artificial na engenharia de software* [Monografia de especialização, Pontifícia Universidade Católica de São Paulo]. Repositório PUC-SP.
- Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.